High-Performance Video in Mac OS X Tiger

Session 207

These are confidential sessions—please refrain from filming or taking pictures.

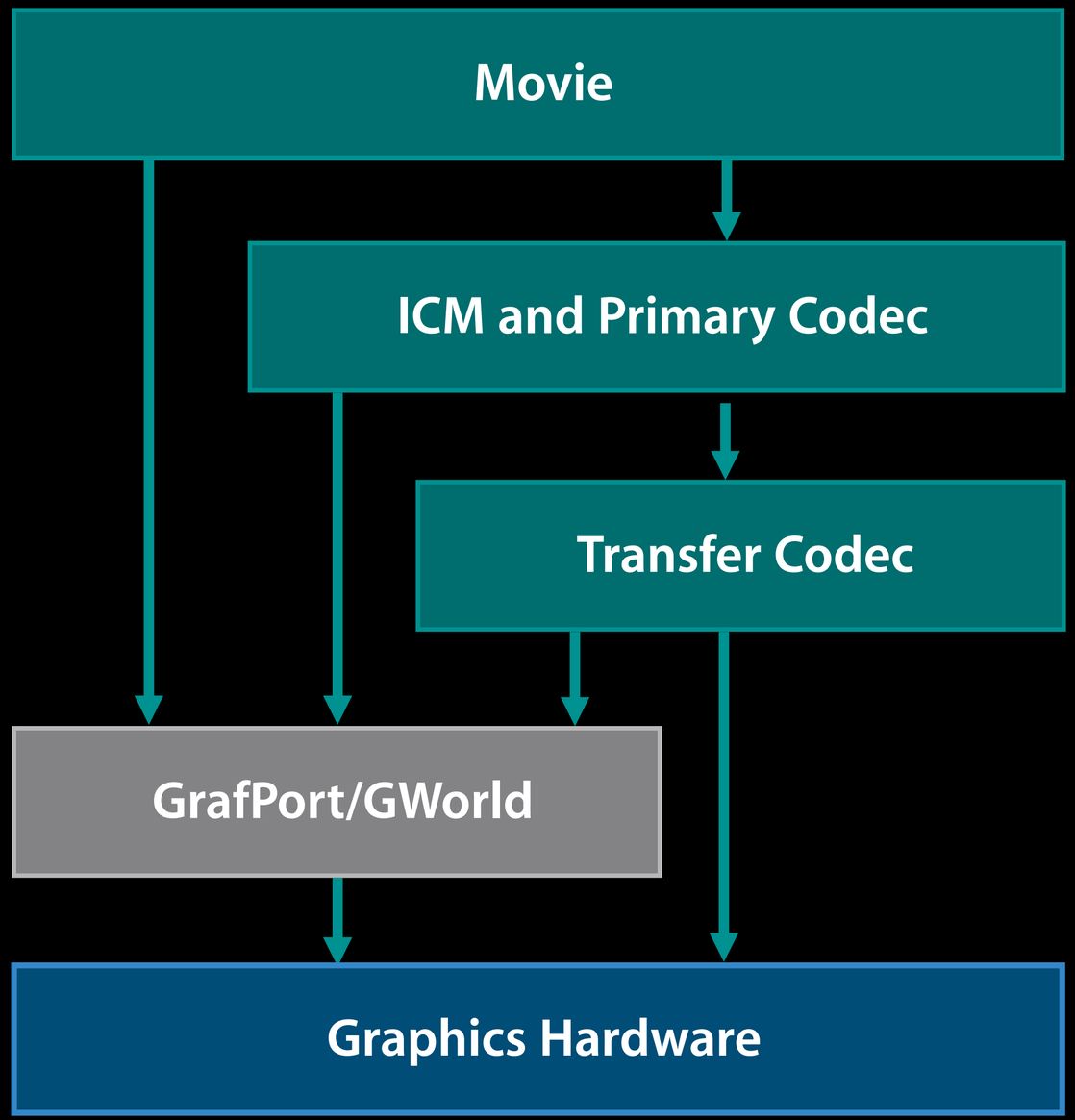
Graphics and Media

High-Performance Video in Mac OS X Tiger

Sean Gies
Video Engineer, QuickTime Video Foundation

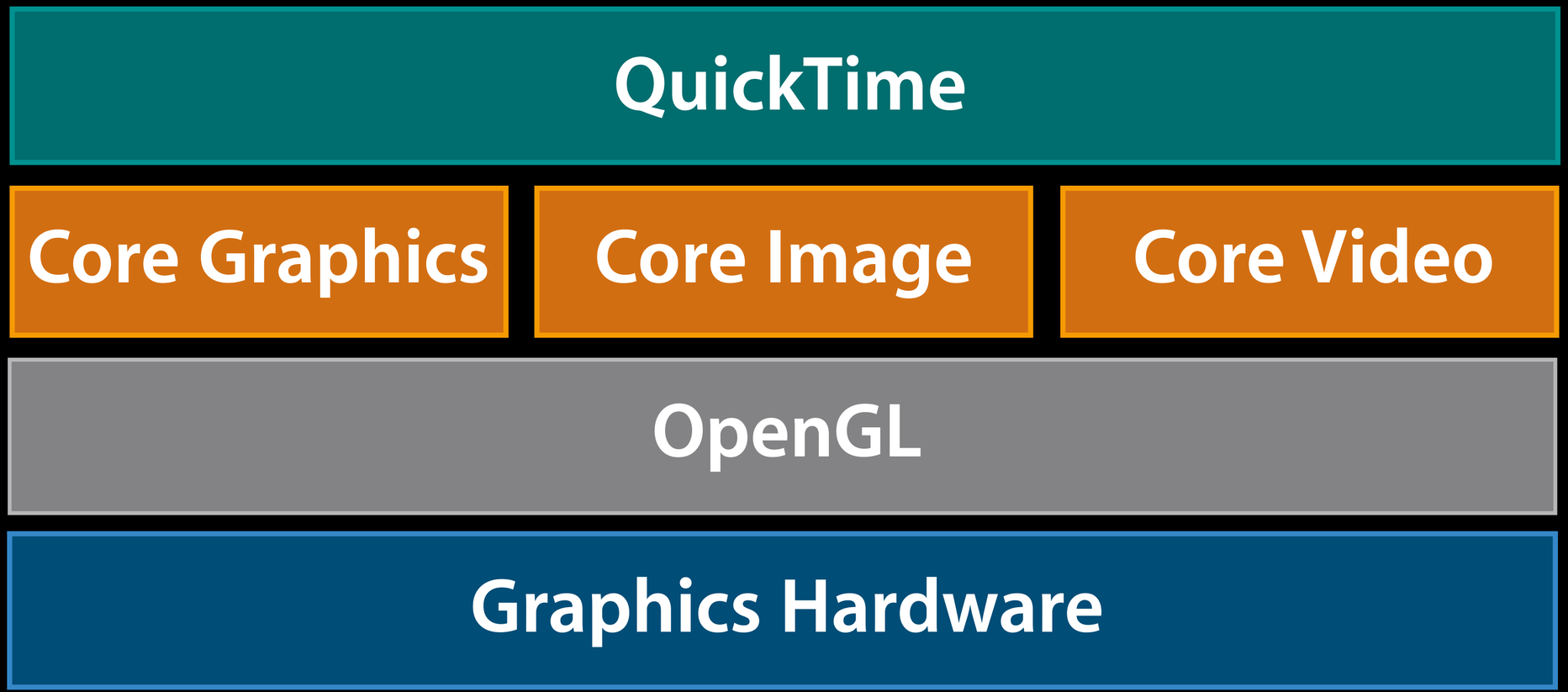
http://developer.apple.com/wwdc2005/

Video Rendering Stack Before Tiger

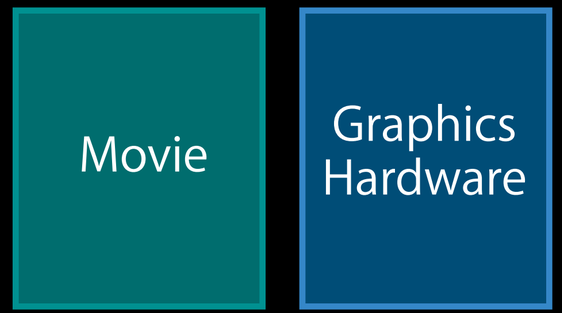


New Video Pipeline in Tiger

- Good integration with OpenGL
- Layered on top of Quartz (Core Video and Core Image)
- Support multiple buffers
- Separated decode and presentation


QuickTime
Core Graphics
Core Image
Core Video
OpenGL
Graphics Hardware

Building the Video Pipeline

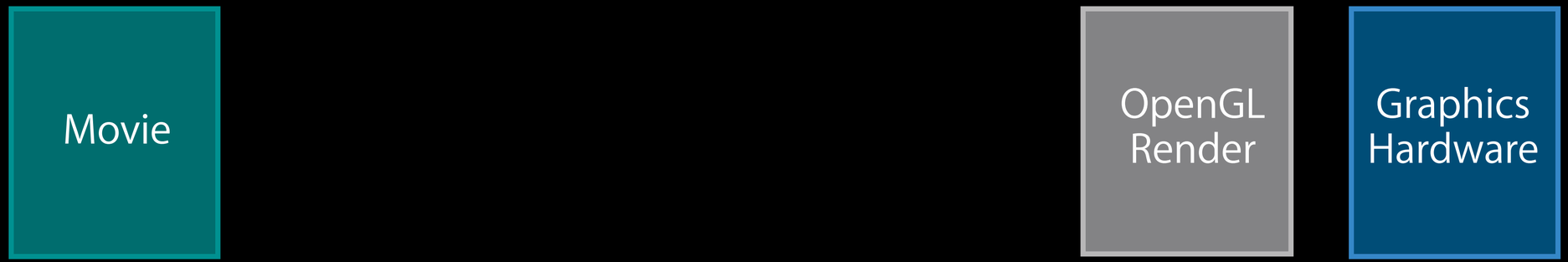


QuickTime Sources and Decodes Frames

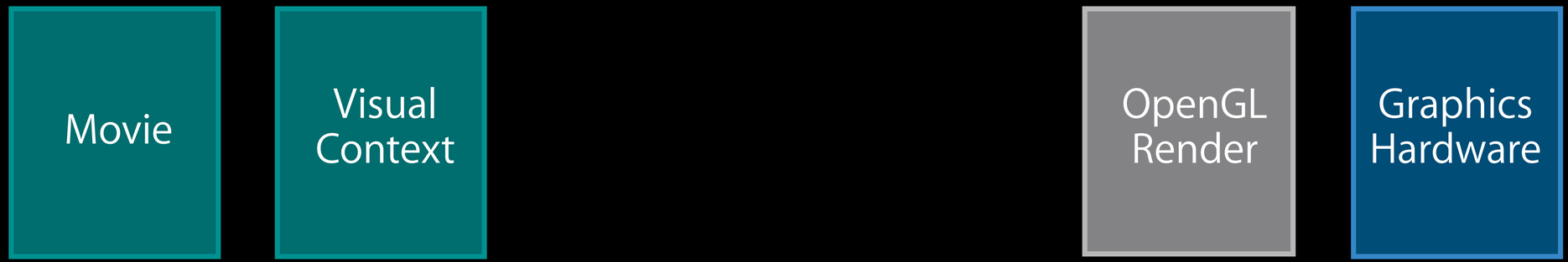
Movie

Graphics Hardware

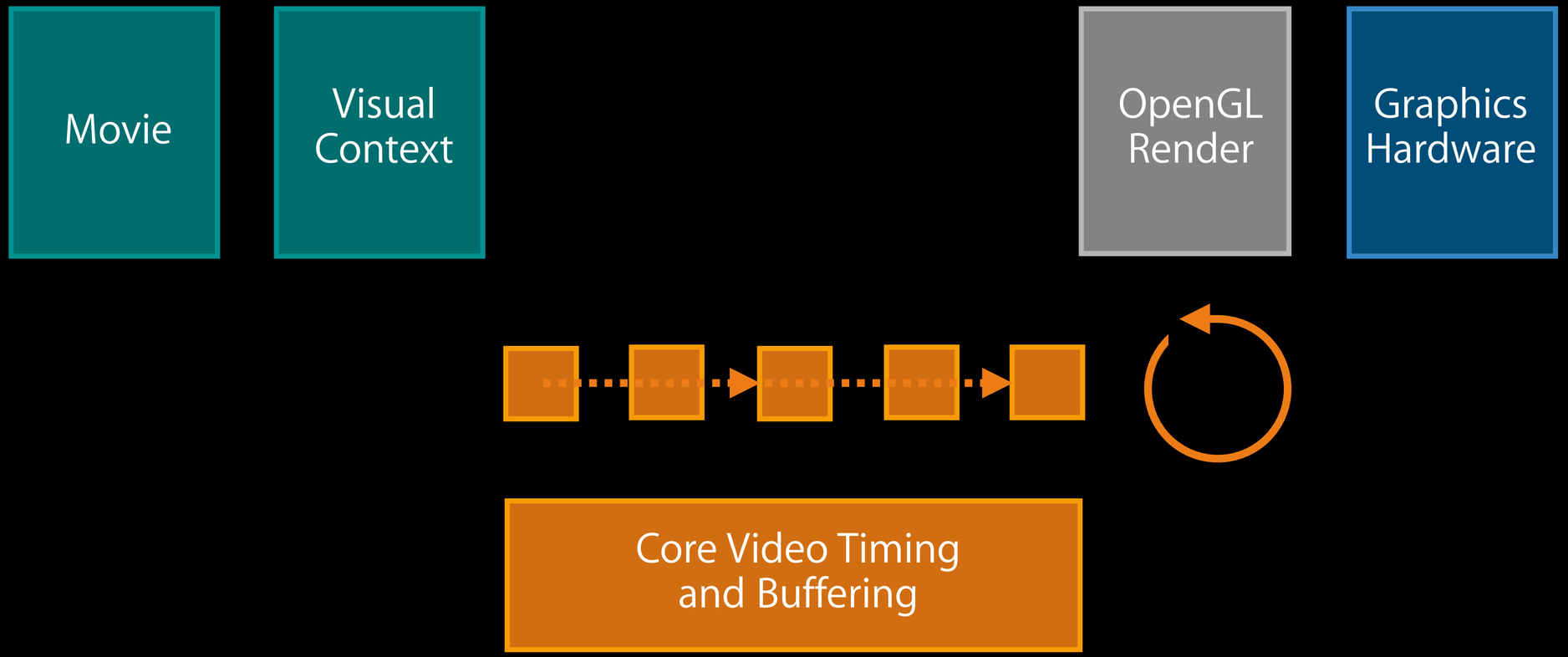
Use OpenGL to Render to Display



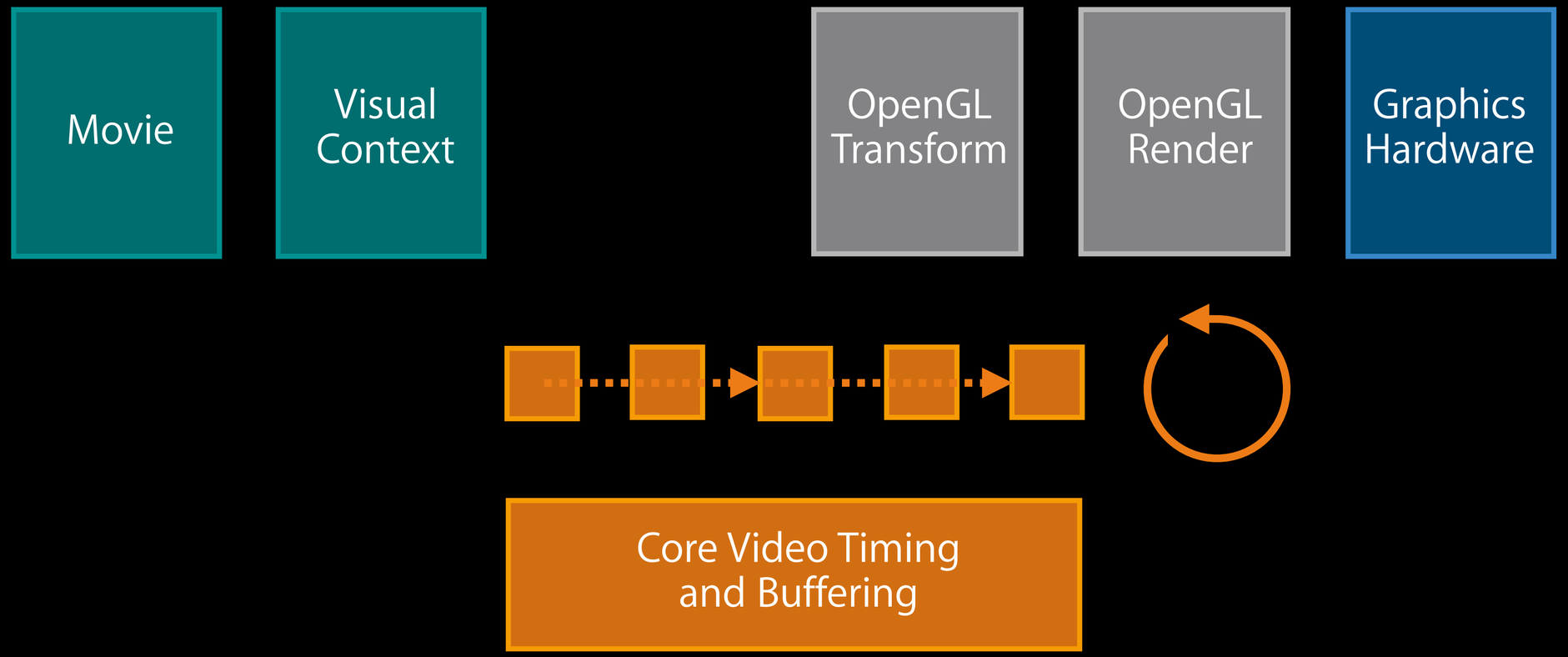
Visual Context Provides Frames From Movie



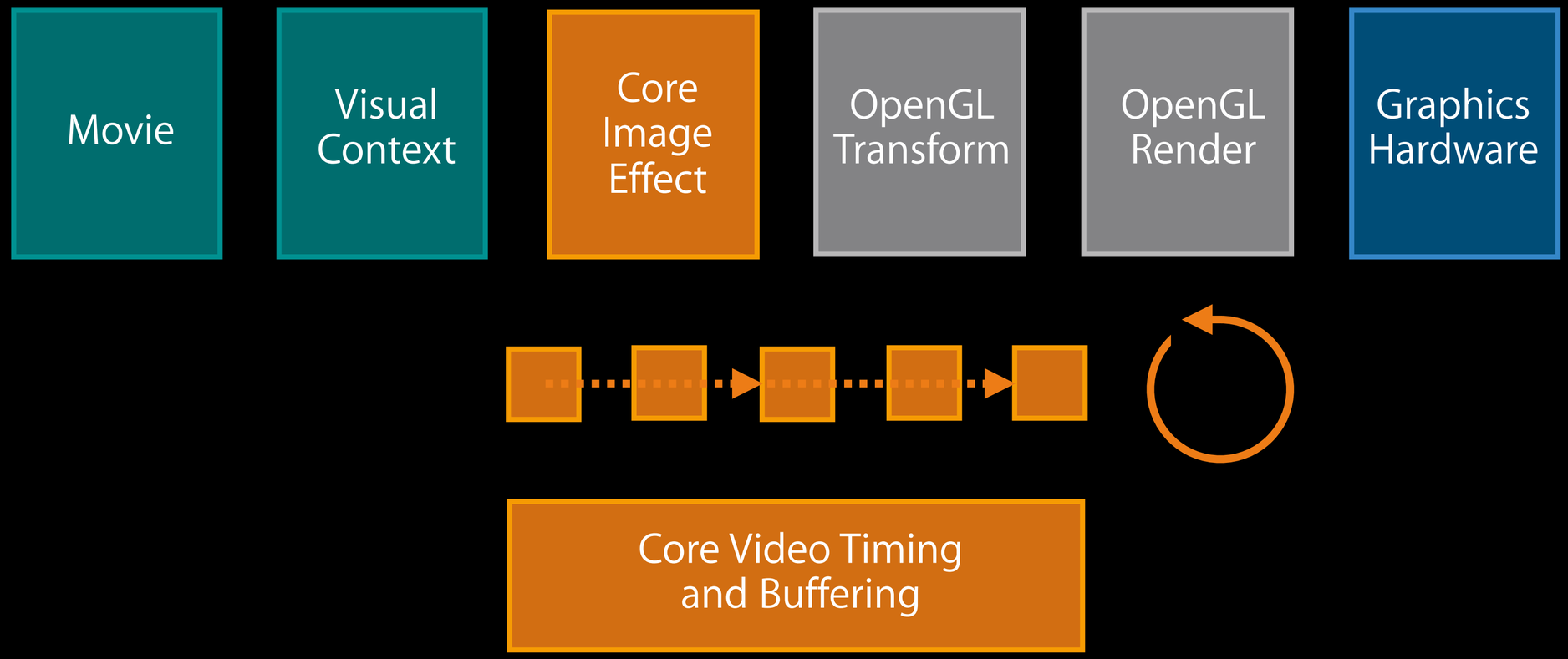
Core Video Provides Timing and Buffer Services



Use OpenGL for Image Transforms



Use Core Image for Image Unit Effects



OpenGL and Core Image Provide Effects

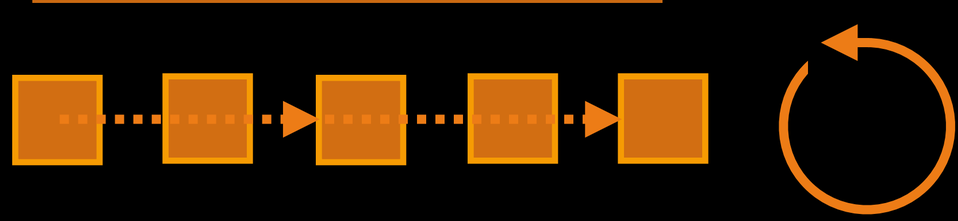
Movie

Visual Context

Application Customized Effects

OpenGL Render

Graphics Hardware



Core Video Timing and Buffering

Advantages

- More pipelining gives higher performance
  - Important for data and CPU intensive formats such as high def H.264
- Your application can easily take advantage of the GPU
  - ColorSync management
  - Filtering and effects
  - Fun new opportunities

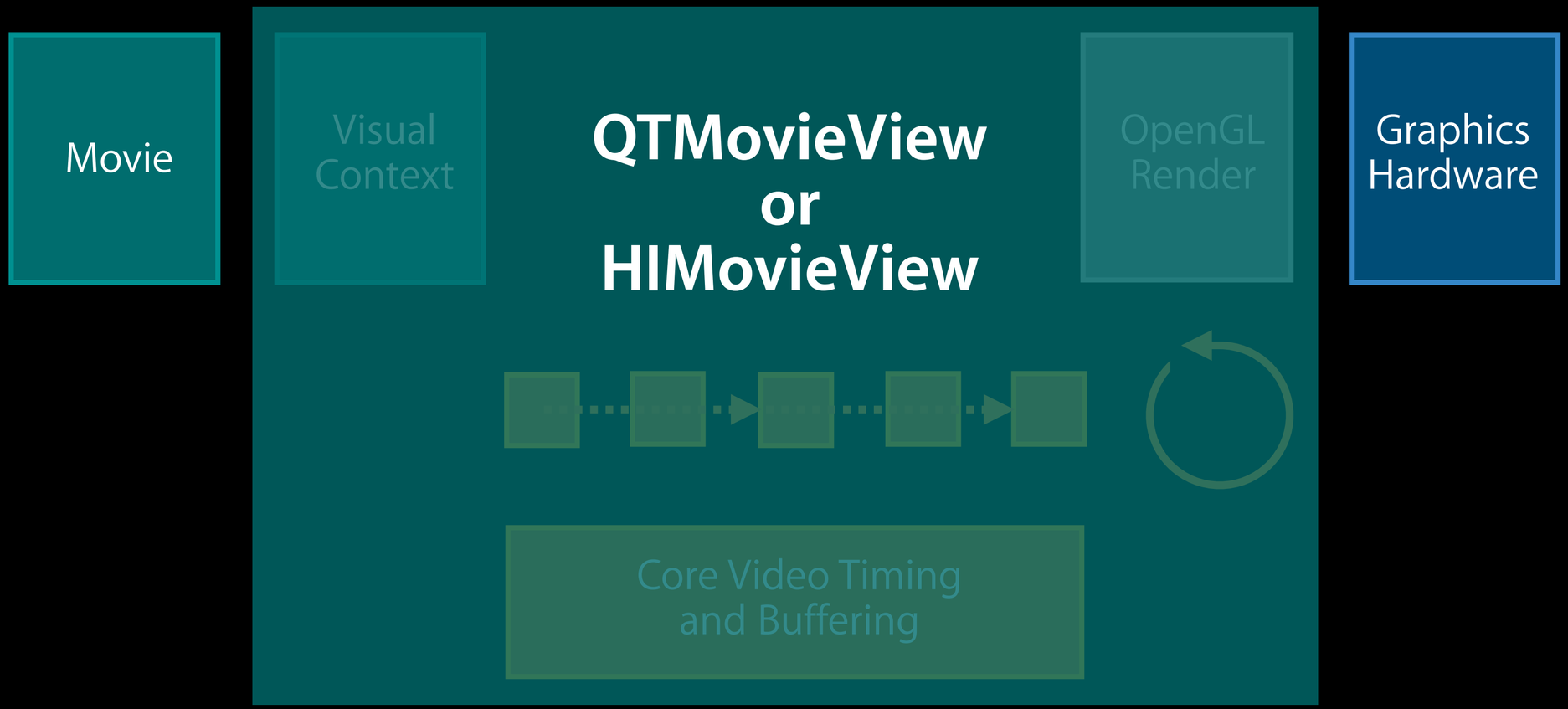
Simple Playback

http://developer.apple.com/wwdc2005/

QuickTime Simple Solutions

- Cocoa
  - QTKit supports drawing using the new video pipeline
    - Use `QTMovie` and `QTMovieView`
      - NSMovie and NSMovieView are deprecated
- Carbon
  - `HIMovieView` also takes advantage of the new pipeline
    - Use instead of CarbonMovieControl on Tiger

Views Handle the Details



Demo

QuickTime Player Leveraging QTKit

http://developer.apple.com/wwdc2005/

Custom Rendering

http://developer.apple.com/wwdc2005/

Customizing the Pipeline

- If you want to…
  - Perform additional image processing
  - Draw over a movie
  - Render within an OpenGL scene
- Can’t just use QTMovieView
- You will need to…
  - Implement some of its behavior
  - Draw with OpenGL and/or Core Image
  - Fall back to GWorlds where necessary

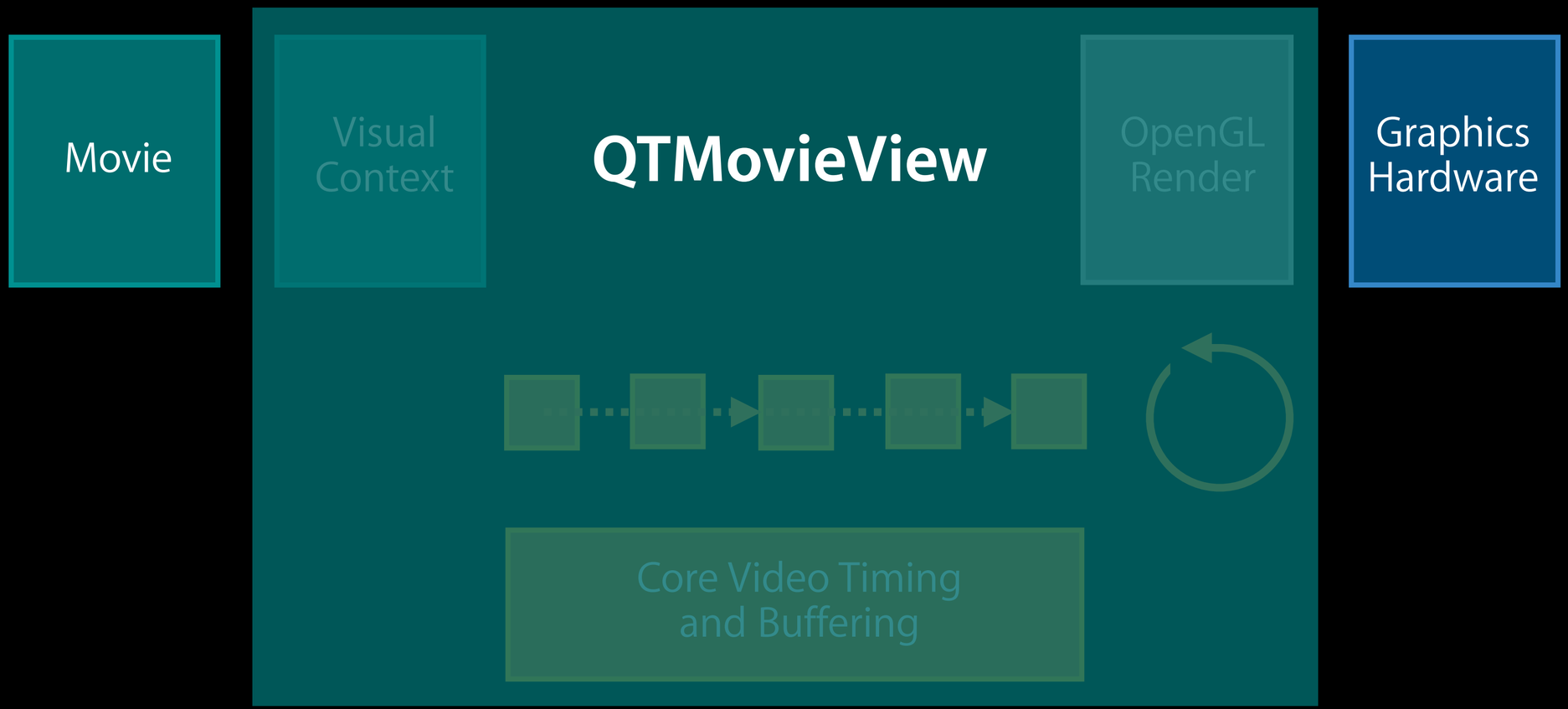
Demo

Customized Rendering Examples

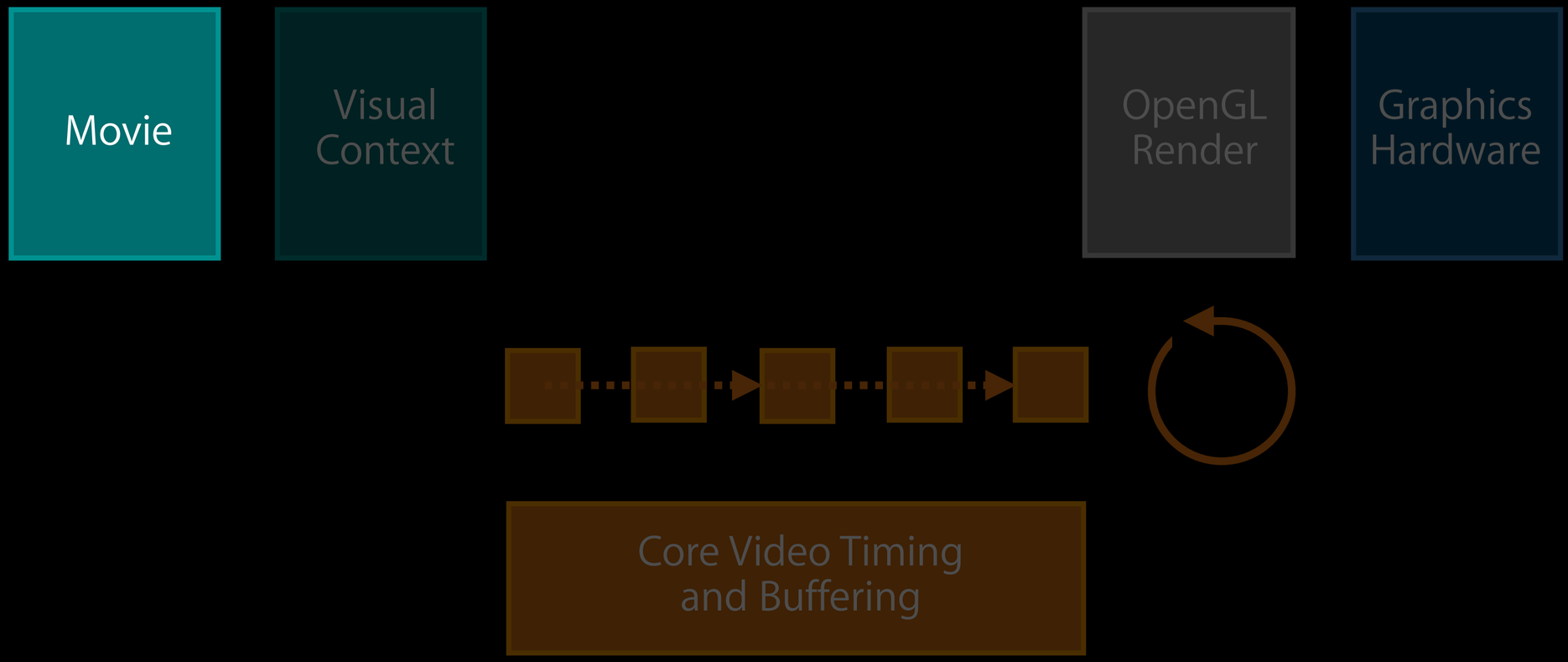
http://developer.apple.com/wwdc2005/

How Does the Pipeline Work?

Under the hood of QTMovieView



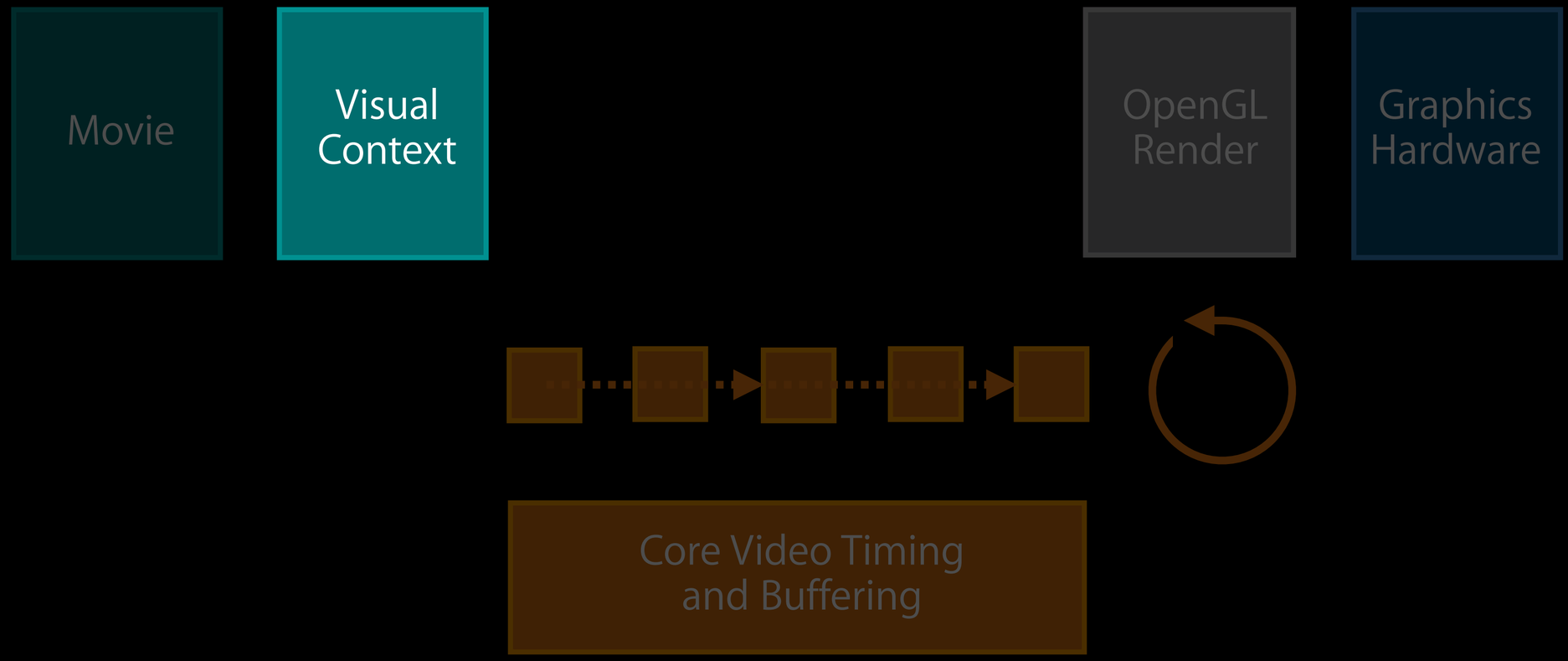
Movie Creation



Movie Creation

- `NewMovieFromProperties`
  - Replaces all other NewMovieXXX calls
    - Options specified via list of properties
    - Results returned through properties
  - Lets you create a movie that doesn't inherit the current GWorld
- QTKit's QTMovie class
  - Uses NewMovieFromProperties internally

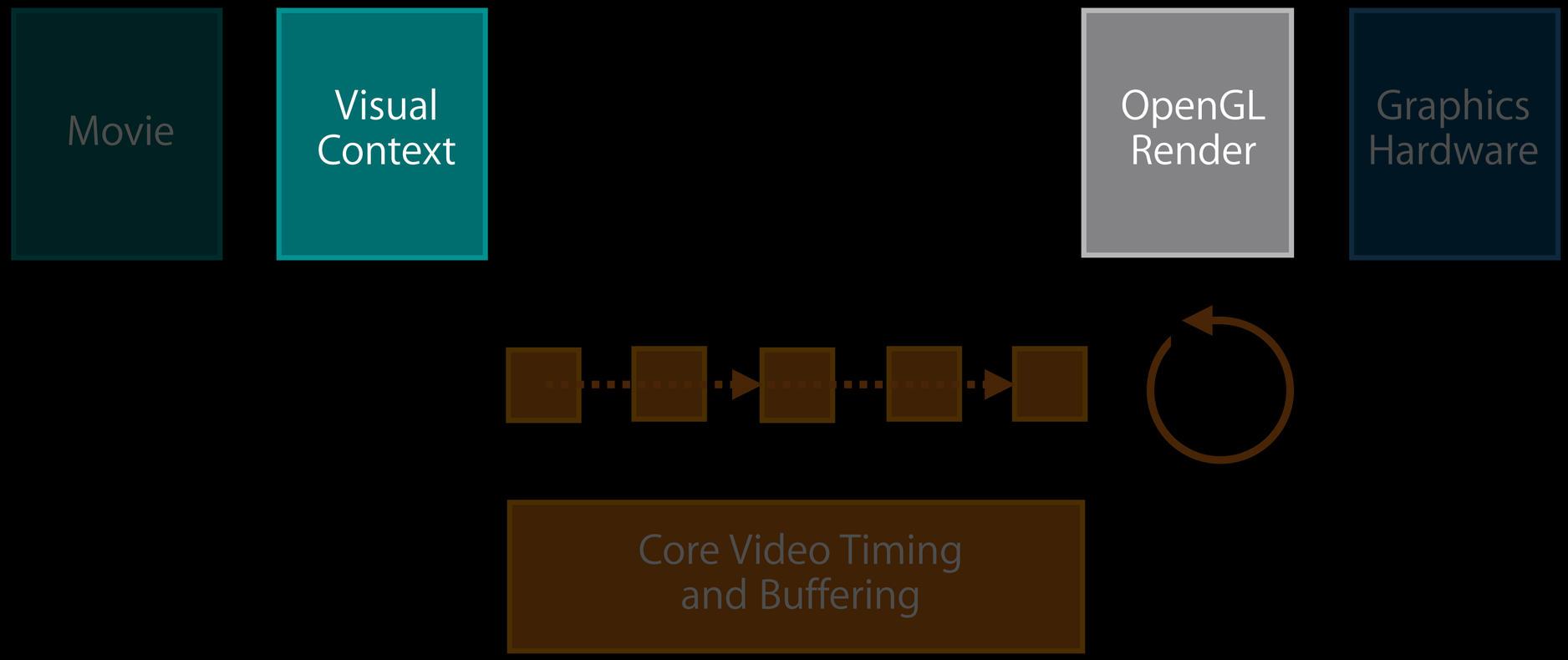
Visual Context



Visual Context

- Abstract drawing destination for QuickTime movies
  - Replaces GWorld
  - Application is responsible for rendering to screen
- Allows for improved QuickTime rendering
  - QuickTime can take advantage of GPU
  - Enables multiple buffers in flight
  - Not restricted to certain movies
  - More asynchronous processing

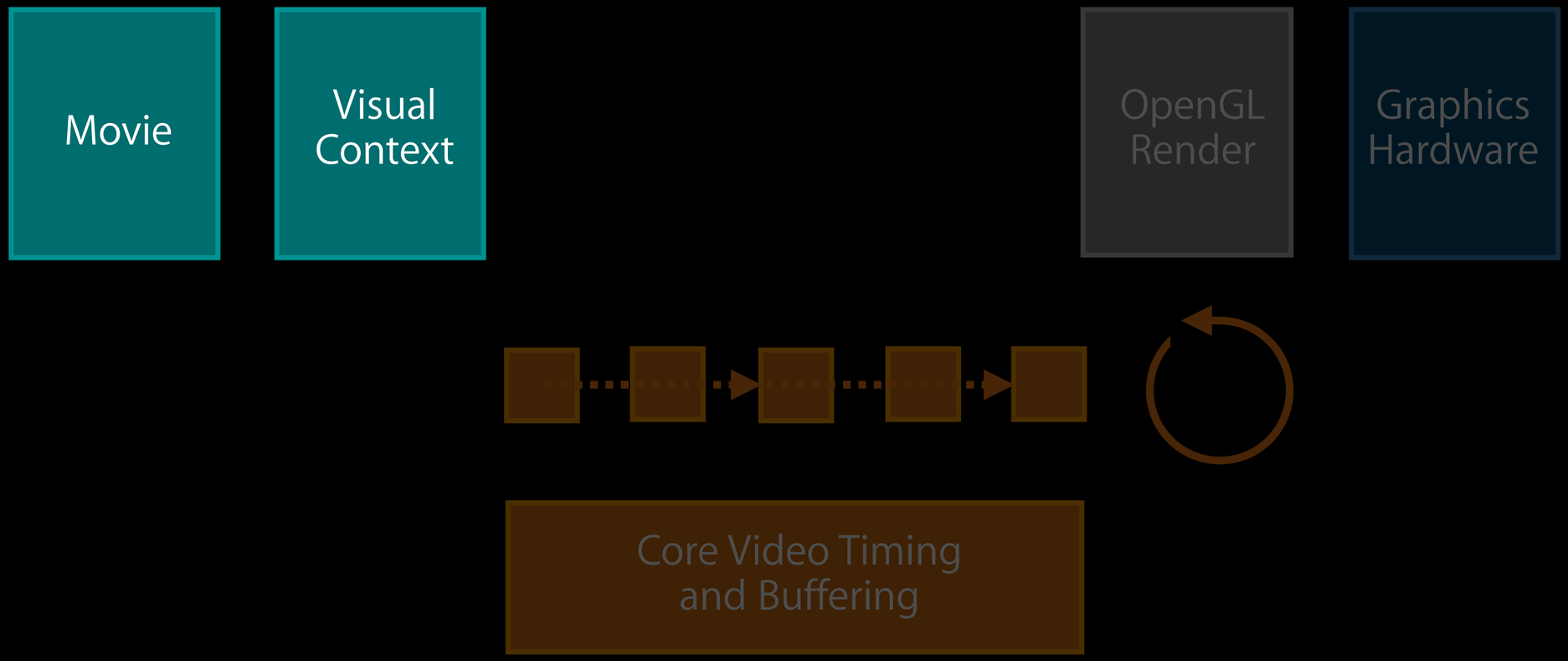
Create a Visual Context



Create a Visual Context

- `QTOpenGLTextureContextCreate` creates a visual context
  - Provides video as OpenGL textures
- Need `CGLContextObj` and `CGLPixelFormatObj`
  - Cocoa apps can get these from NSOpenGLContext and NSOpenGLPixelFormat
  - Carbon apps can get these from AGLContext and AGLPixelFormat
- Can fail due to insufficient graphics hardware

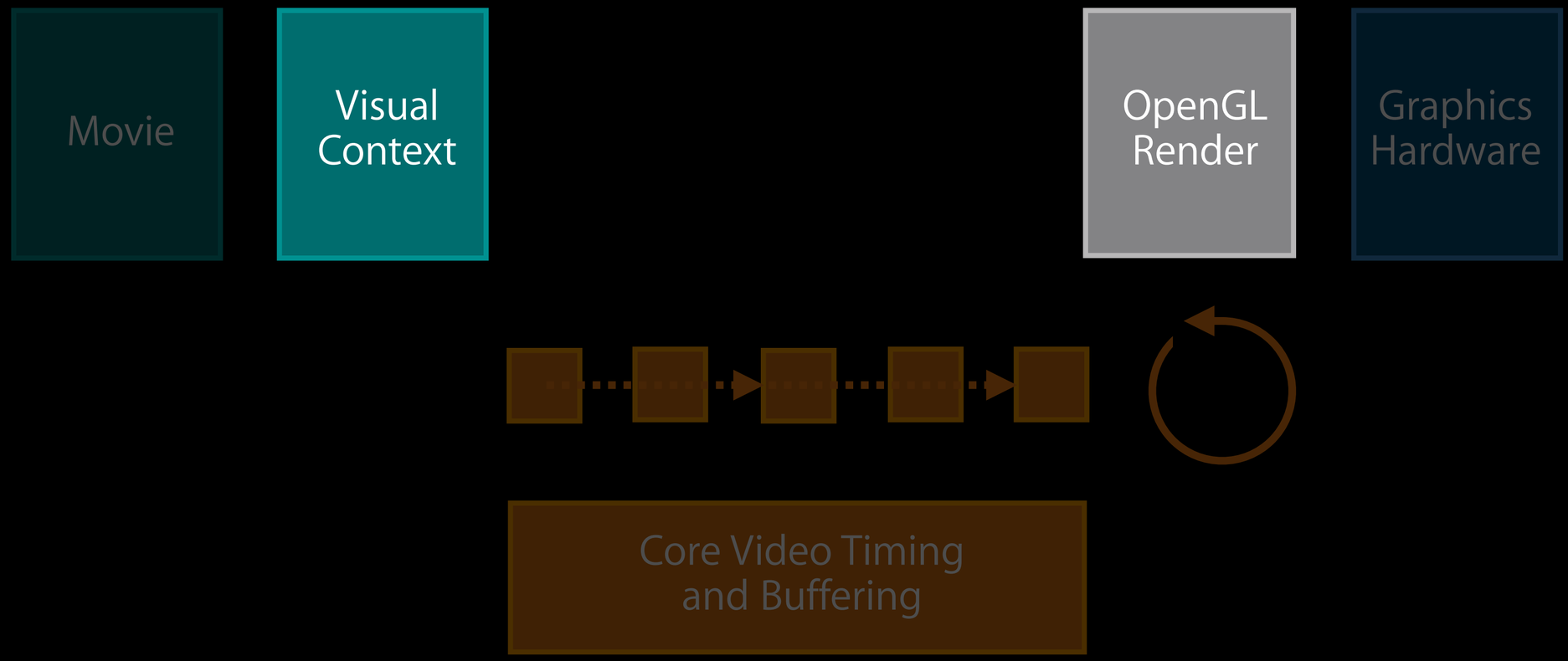
SetMovieVisualContext



SetMovieVisualContext

- Replaces `SetMovieGWorld`
- Points a movie to a visual context
  - Fails if visual context is already connected elsewhere
- `SetMovieVisualContext` to NULL
  - Tells a movie not to draw (unlike `SetMovieGWorld`)
  - Used to switch between Visual Contexts and GWorlds
  - Forces movie to release the visual context (important for compatibility reasons)

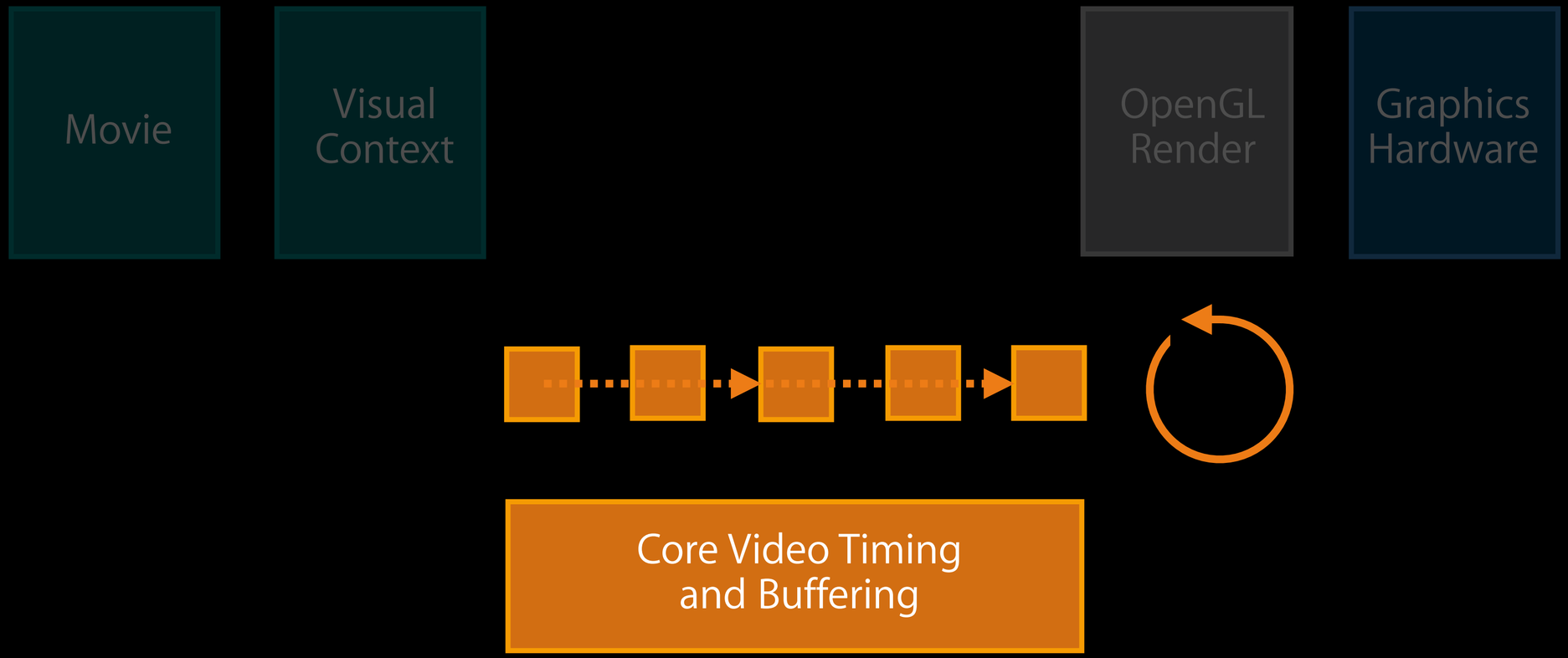
Getting the Images



Getting the Images

- `QTVisualContextIsNewImageAvailable`
  - Query for a new video frame
- `QTVisualContextCopyImageForTime`
  - Pulls the frame for a given time
- `QTVisualContextTask`
  - For thread-safe OpenGL housekeeping

Core Video Overview



Core Video Overview

- Buffer management
  - Provide common buffering model for multiple frameworks
  - Assist with resource management/re-use
- Display synchronization
  - Analogous to Core Audio

Core Video Buffers

- Core Foundation-based
- Abstract CVBuffer and CVImageBuffer types
  - Common buffer model for QuickTime and Core Video
- Buffer attachments
- Multiple concrete CVImageBuffer types in Tiger
  - CVPixelBuffer
  - CVOpenGLTexture
  - CVOpenGLBuffer

CVOpenGLTexture

- Represents an OpenGL texture
- Internally handles the tricky parts of creating OpenGL textures from pixel buffers
- Provides texture target, name, size, etc.
- Use provided texture coordinates!
  - `CVOpenGLTextureGetCleanTexCoords`

Core Video Display Link

- Assists with driving visual context pull model
  - Matches update requests to display refresh rate
  - Provides time stamp information for the next display time
    - `CVTimeStamp` similar to AudioTimeStamp
- Single output callback
  - Dedicated I/O thread
- Created from a set of displays

Rendering the Images: Core Image vs. OpenGL

- Core Image
  - Might not be available
  - Color Matching built in
  - Effects
  - Supports transforms and compositions too
- OpenGL
  - Broader hardware support
  - More general purpose and most efficient
  - More code to write

Rendering the Images: Color Management

- QuickTime media can be tagged with color information
  - Apps have been tagging content for a while
  - See Ice Floe dispatch 19
    - http://developer.apple.com/quicktime/icefloe/dispatch019.html
- Color Management uses the same model as Core Image
  - Separate working and output color spaces
  - ColorSync is used to match colors
- Little code required

Sample Code: VideoViewer

- Uses QTMovie for movie management
- Doesn’t use QTMovieView
- Implements rendering via
  - OpenGL
  - Core Image
  - GWorld fallback

Demo and Code

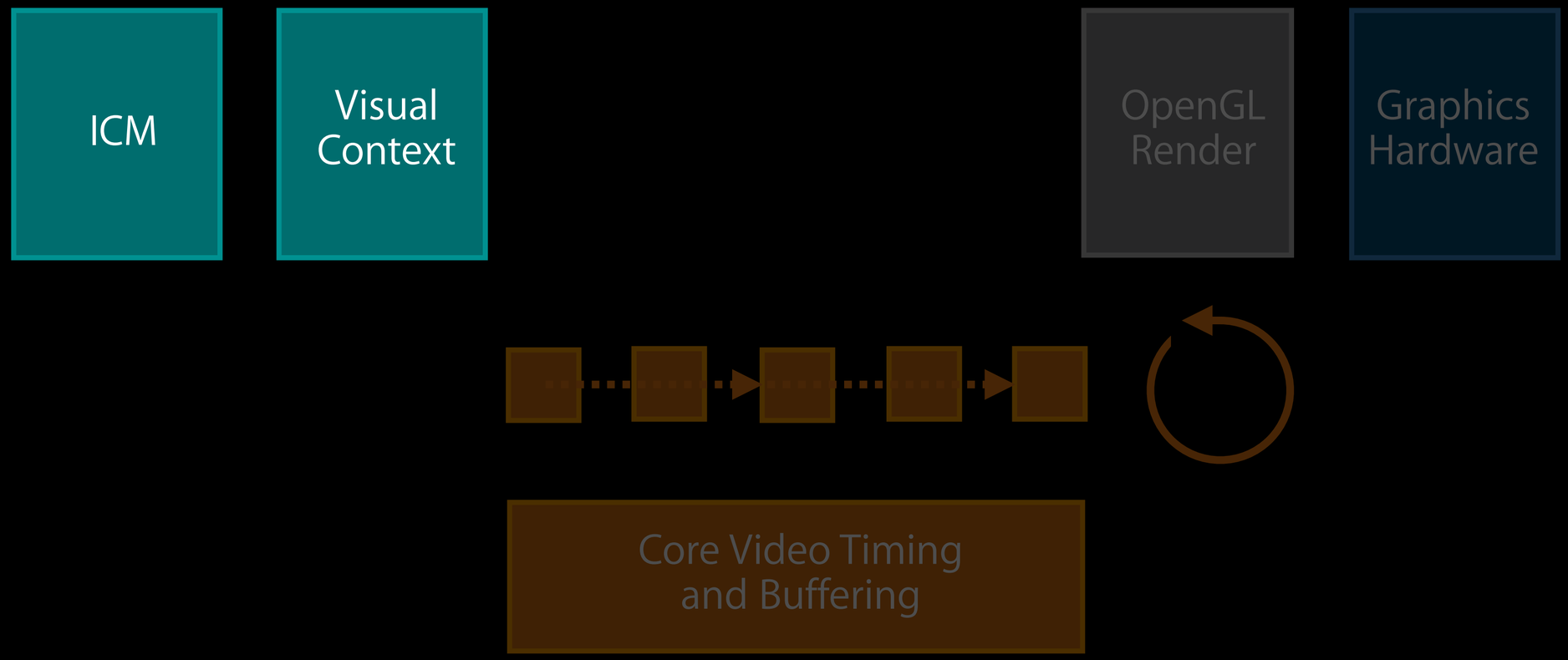
VideoViewer

http://developer.apple.com/wwdc2005/

Other Uses of Visual Contexts

http://developer.apple.com/wwdc2005/

ICM Drives Visual Contexts Too



ICM Drives Visual Contexts Too

- Use `ICMDecompressionSessionCreateForVisualContext`
  - Access at a lower level than Movie
  - Use for compressed or uncompressed frames
    - e.g. captured data from sequence grabber, rendered previews

CVPixelBuffer

- Used for memory-based pixel data
- Used by QuickTime for internal buffers
- Foundation for multi-buffer ICM APIs

Pixel Buffer Visual Context

- `QTPixelBufferContextCreate`
  - Provides CVPixelBuffers from a movie
  - Used for non GPU-based rendering or extraction
  - Options to control size, pixel format, etc.

Leveraging Core Image

Frank Doepke
Software Engineer and Pixel Pipeline Pusher, Quartz Effects

http://developer.apple.com/wwdc2005/

Movie

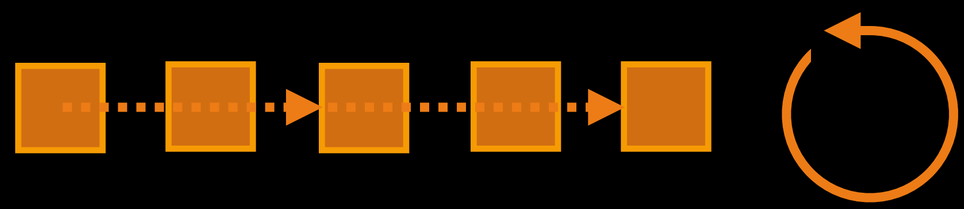
Visual Context

Core Image Effect

OpenGL Transform

OpenGL Render

Graphics Hardware



Core Video Timing and Buffering

Movie

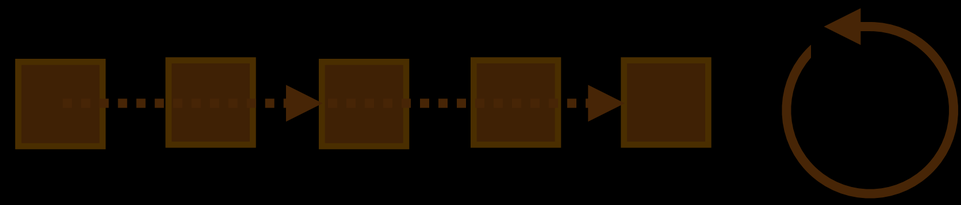
Visual Context

Core Image Effect

OpenGL Transform

OpenGL Render

Graphics Hardware



Core Video Timing and Buffering

Adding Core Image Effects to Playback

- Core Image bling
  - Provides 100+ filters
  - Fast processing on the GPU
- Going from a CVImageBuffer to a CIImage
- Integrating CI with an NSOpenGLContext
- Filters can be chained
  - Chain folded at rendering time

Sample Code: CIVideoDemoGL Episode II

- Integrates Core Image with Core Video, OpenGL and QuickTime
  - CIContext from NSOpenGLContext
  - `CIImageFromCVImageBuffer` to create a CIImage from a QuickTime frame
    - Don’t use imageWithTexture
- Creates a filter pipeline
  - Setting up filters
  - Use CIComposite filters for compositing

Demo and Code

CIVideoDemoGL—adding effects

http://developer.apple.com/wwdc2005/

Exporting From the GPU

- CIContext `createCGImage:`
  - Will read back from CIContext
  - Great if all you need is a CGImage
- `glReadPixels`
  - Will read back from OpenGL context
  - Not specific to Core Image
- Other more advanced techniques

Maximizing OpenGL Performance	Session 214 Thursday 2:00PM

Sample Code: CIVideoDemoGL

- Going from CI to ImageIO to a file
- Going from the GL context to a QuickTime movie
  - Using `glReadPixels` to read from the OpenGL context
  - Using `MovieExportFromProcedures` for QuickTime export

Demo and Code

CIVideoDemoGL—export and readback

http://developer.apple.com/wwdc2005/

Customizing Your OpenGL Rendering

http://developer.apple.com/wwdc2005/

Movie

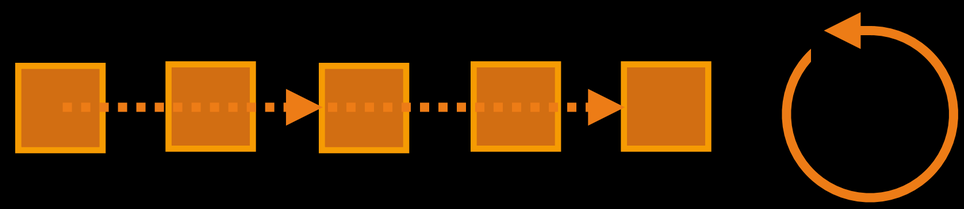
Visual Context

Core Image Effect

OpenGL Transform

OpenGL Render

Graphics Hardware



Core Video Timing and Buffering

Movie

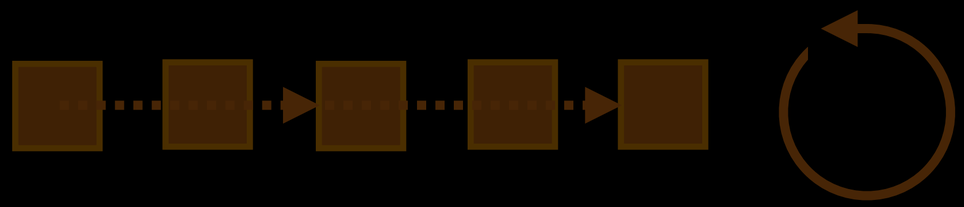
Visual Context

Core Image Effect

OpenGL Transform

OpenGL Render

Graphics Hardware



Core Video Timing and Buffering

Always Wear Thread-Safety Glasses

- OpenGL contexts are not re-entrant
- All OpenGL calls need to be protected
  - Use pthread locks or NSLocks
  - A shared OpenGL context
  - CGLLockContext/CGLUnlockContext (in Tiger)
- Use locks when accessing QTOpenGLTextureContexts
  - IsNewTextureAvailable is thread-safe
- Note to AppKit users
  - Override `update` method on NSOpenGLViews to lock view around `[super update]`

More Thread-Safety

- Use the OpenGL Profiler to detect thread conflicts
  - Check the Maximizing OpenGL lab and session

Maximizing OpenGL Performance	Session 214 Thursday 2:00PM

Sample Code: LiveVideoMixer

- Revenge of LiveVideoMixer
  - Sample code has been revised
  - Three movies playing to one surface
  - Uses OpenGL for rendering, effects and compositing
- What is new—readback using AVC Video Services
  - Leveraging glReadPixels for DV FireWire output
    - Would work for Core Image or OpenGL

FireWire	Session 504 Tuesday 2:00PM

Demo and Code

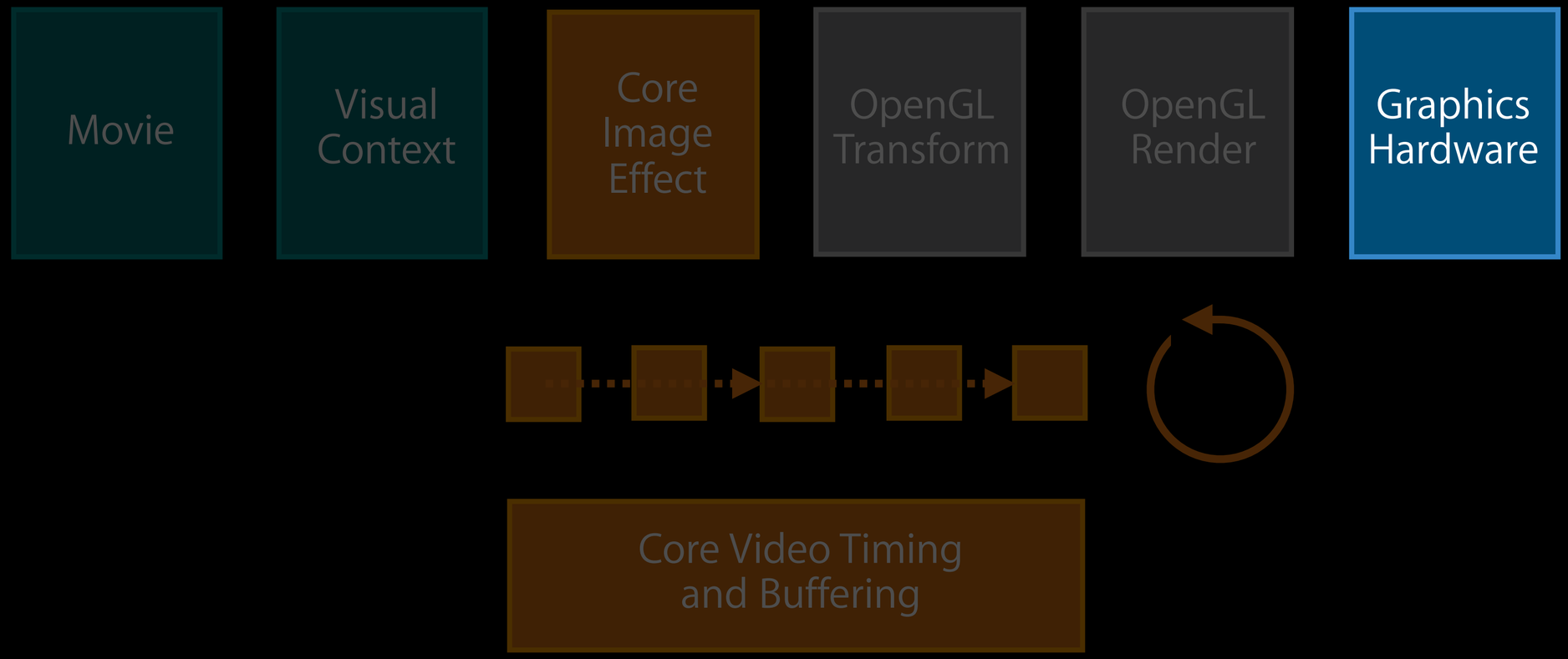
LiveVideoMixer and FireWire Output

http://developer.apple.com/wwdc2005/

Demo Summary

- Compositing
  - glBlend together with alpha
  - Masking using Multitextures
- FireWire out
  - glReadPixels to read back the pixels from the graphics hardware
  - Flip-flopping the coordinate system
  - Compress into DV using QuickTime
  - Send to the FireWire device using the AVC Video Services

Caveat Implementor

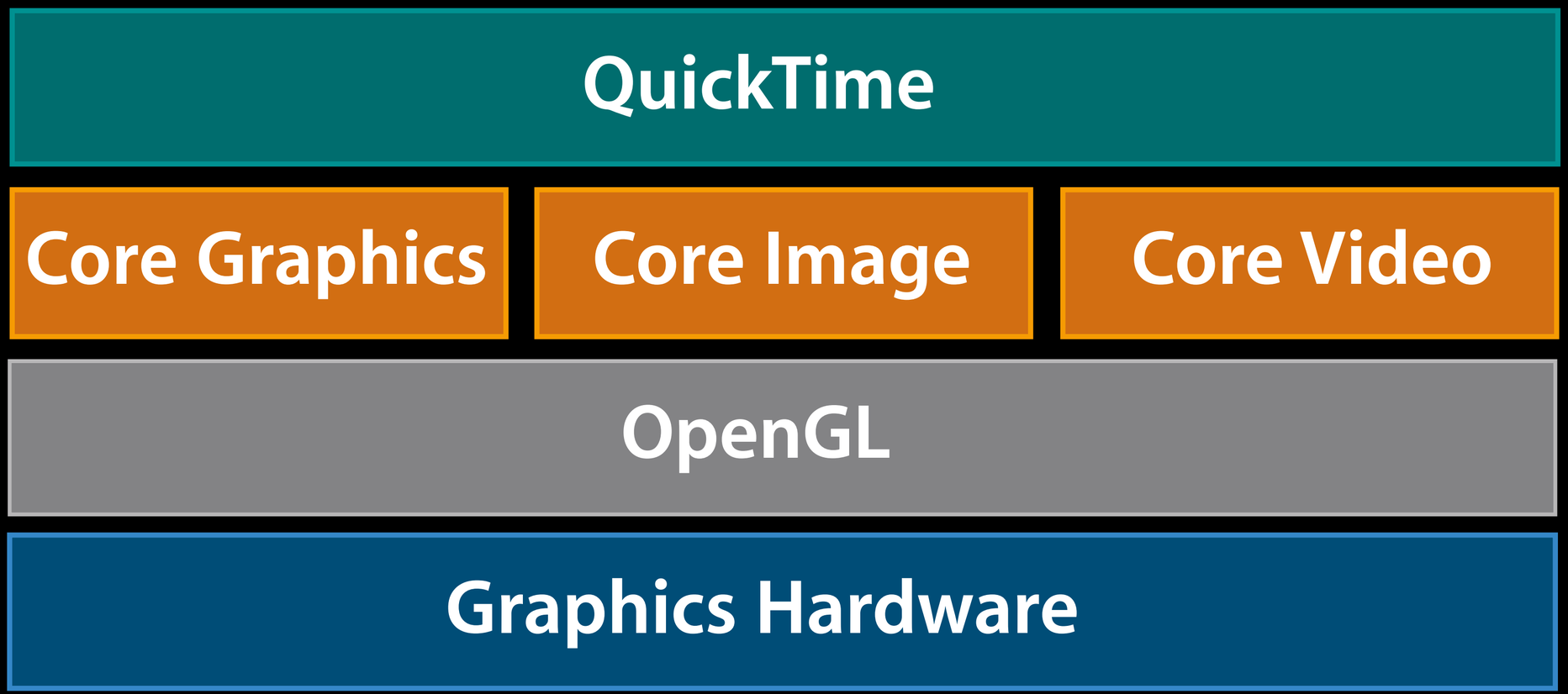


Caveat Implementor

- Subject to limitations of graphics hardware
  - OpenGL
    - Quartz Extreme capable on all displays
    - Texture and drawable size limits
      - Resize movie box to reduce requirement
  - Core Image
    - Always available on Tiger
    - GPU accelerated on programmable GPUs
- Watch for thread safety!

Summary

http://developer.apple.com/wwdc2005/


QuickTime
Core Graphics
Core Image
Core Video
OpenGL
Graphics Hardware

New Video Pipeline in Tiger

Movie

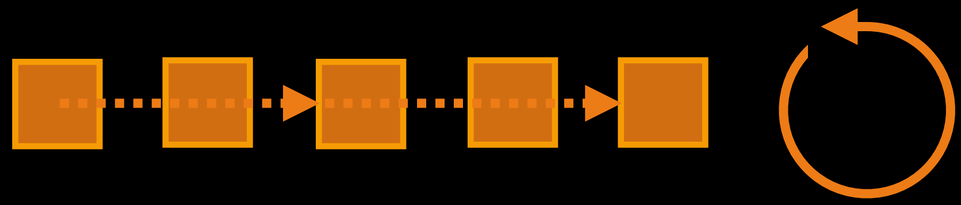
Visual Context

Core Image Effect

OpenGL Transform

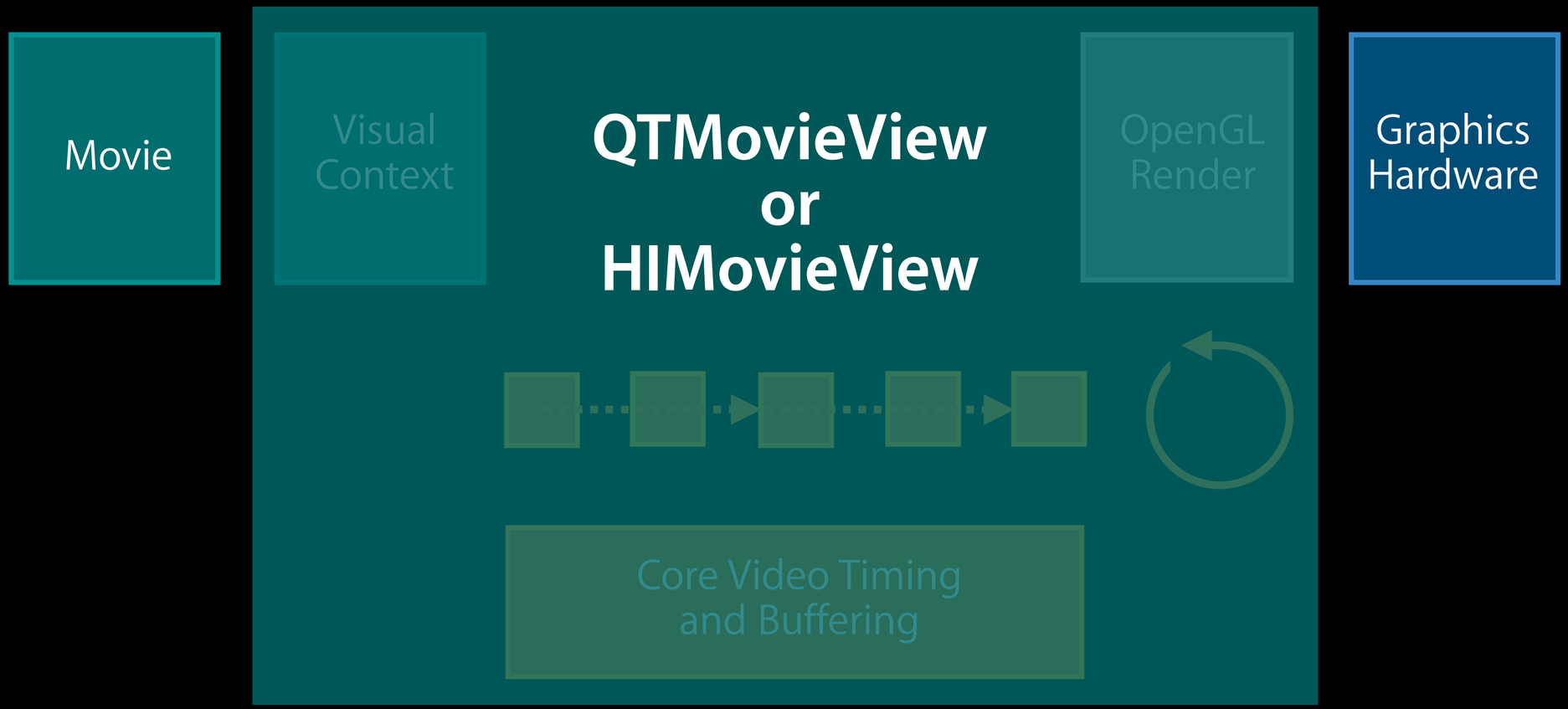
OpenGL Render

Graphics Hardware

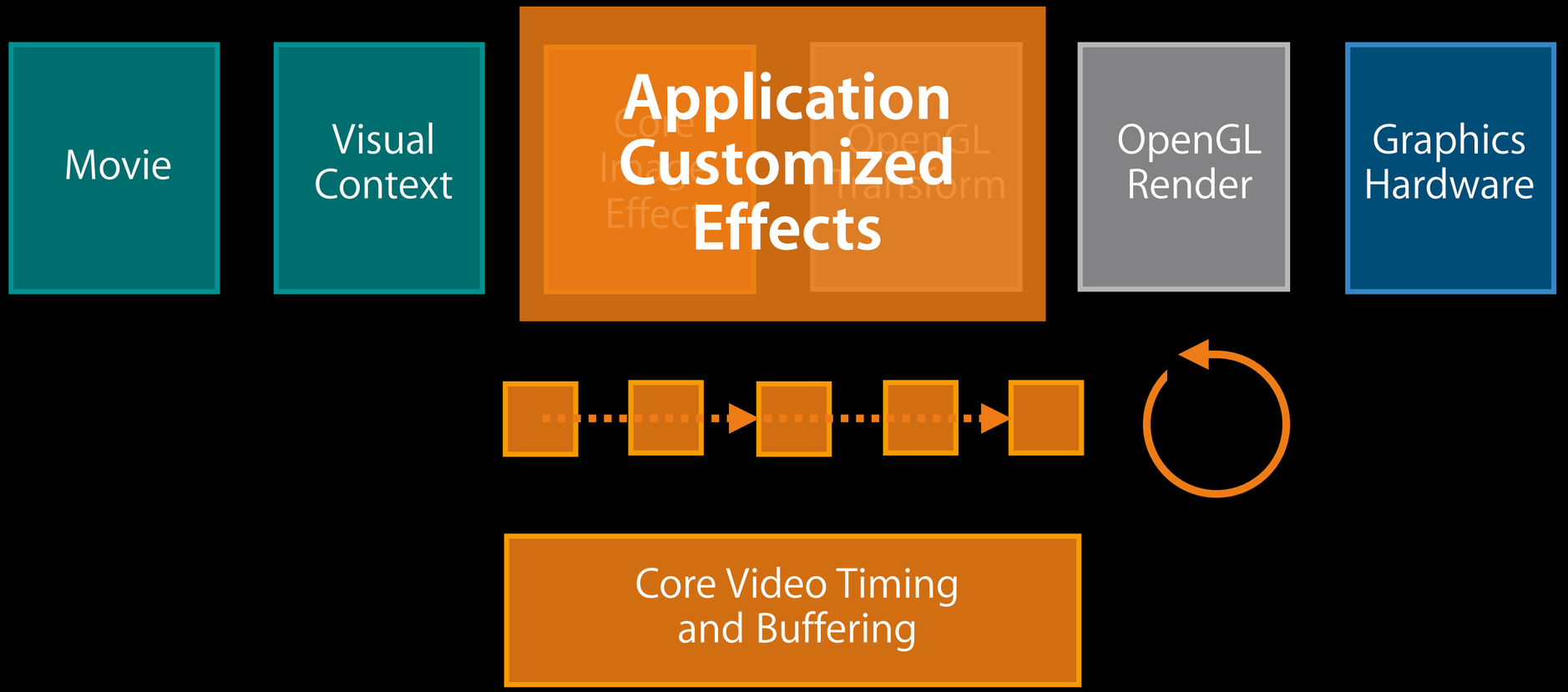


Core Video Timing and Buffering

Use Movie Views for High-Level Access



Customize Your Applications



More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

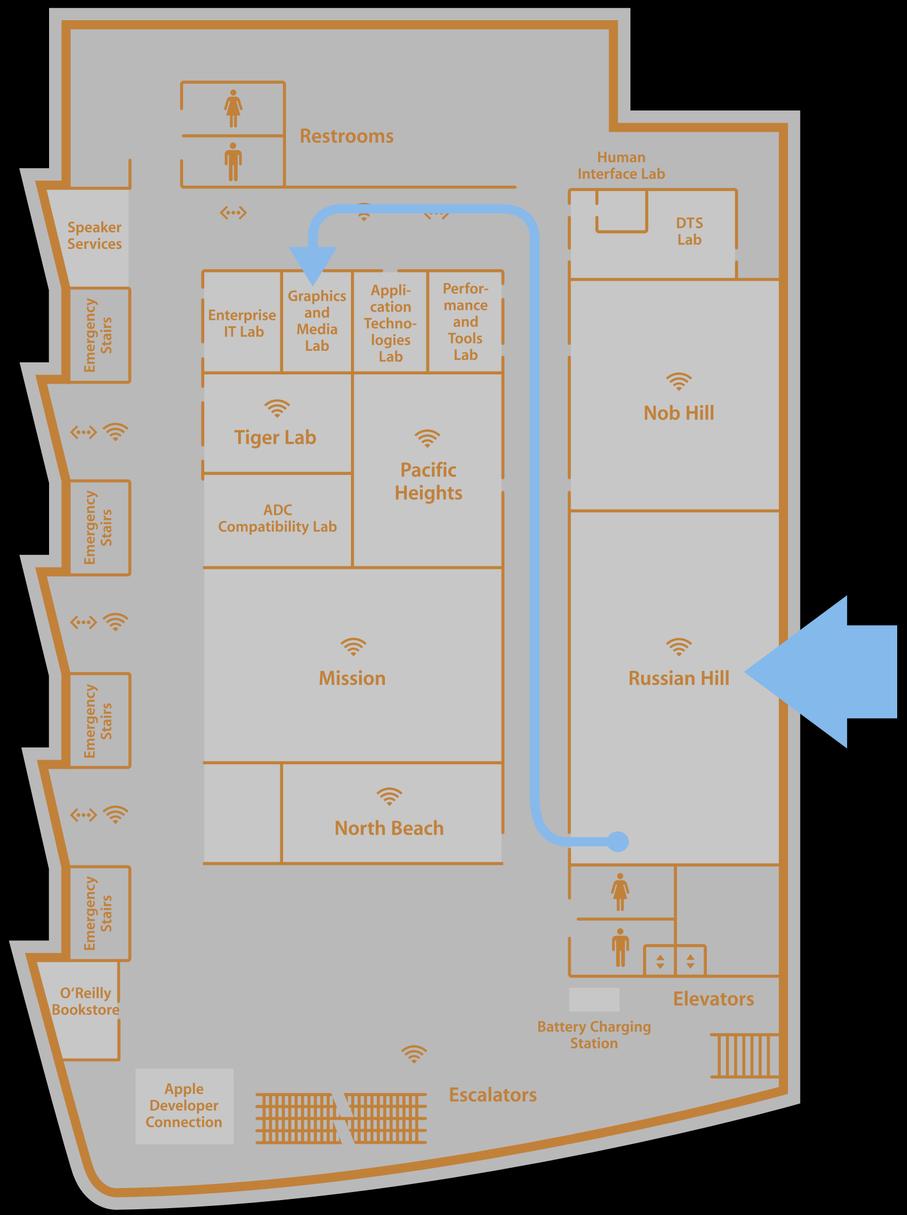
Related Sessions

FireWire	Session 504 Tuesday 2:00PM
Supporting Advanced Video Formats	Session 208 Wednesday 2:00PM
Mac OS X Color Image Management Explained	Session 209 Wednesday 2:00PM
Maximizing OpenGL Performance	Session 214 Thursday 2:00PM

Feedback Forums

Graphics and Media	Feedback Forum Friday 2:00PM
QuickTime	Feedback Forum Friday 3:30PM

Visit Us in the Graphics and Media Lab!



Sample Code

- VideoViewer
- CIVideoDemoGL
- LiveVideoMixer
- VisualContextSample
- QTCoreVideo101
- QTCoreImage101

Q&A

Tim Cherna
Engineering Manager, QuickTime Video Foundation

WWDC2005

Worldwide Developers Conference